Claim Amendments

Please amend the claims as indicated:

1. (original) A method for implementing subroutine calls and returns in a computer system comprising the following steps:

A) converting a sequence of input language (IL) instructions into a corresponding sequence of output language (OL) instructions;

B) executing the OL instructions;

C) for each call to an IL subroutine made from an IL call site in the IL instruction sequence:

i) storing a call site IL return address R_{call} on a stack;

ii) calculating a first index by evaluating a function with P as an argument, where P is a procedure entry address of the subroutine;

iii) storing a corresponding OL return address in a return target cache at a location indicated by the first index;

iv) executing an OL subroutine translation of the called IL subroutine;

D) upon completion of execution of the OL subroutine translation,

i) in a launch block of instructions, retrieving an OL target address from the return target cache at the location indicated by a second index; and

ii) continuing execution beginning at the OL target address.

2. (original) A method as in claim 1, further including the following steps:

determining whether a predicted IL return address R_{pred} is the same as an actual IL return address R_{actual} fetched from the stack and, if it is not, transferring execution to a back-up OL return address recovery module; and

in the back-up OL return address recovery module, establishing the OL return address using a predetermined, secondary address recovery routine.

3. (original) A method as in claim 2, in which there is a plurality of IL call sites, further including the following steps:

translating each IL call site into a corresponding OL call site;

generating a confirm block of instructions corresponding to each OL call site;

upon execution of any confirm block of instructions:

comparing the actual IL return address R_{actual} with the predicted IL return address R_{pred};

if R_{actual} is equal to R_{pred}, continuing execution of the OL instructions following the OL call site; and

if R_{actual} is not equal to R_{pred}, transferring execution to the back-up return address recovery module.

4. (original) A method as in claim 3, in which only a single scratch register is used for the in the launch and confirmation blocks of instructions.

5. (currently amended) A method as in claim 3, in which:

the return target cache is an array having a plurality of elements;

the function maps IL return addresses with a uniform probability distribution over at least a subset of the return target cache;

equality and inequality between R_{actual} and R_{pred} are defined as a hit and a miss, respectively;

further including the following steps:

calculating a return success measure as a function of the frequency of occurrence of hits relative to the frequency of occurrence of misses;

adjusting the number of elements in the return target cache according to a function of the return success measure.

6. (original) A method as in claim 2, in which the return target cache is an array having a plurality of elements, further including the step of initializing the return target cache by storing in each element a beginning address of the back-up return address recovery module.

7. (currently amended) A method as in claim 1, in which:
the return target cache is an array having a plurality of elements; and
the function maps IL procedure entry addresses with a uniform probability distribution over at least a subset of the return target cache.

8. canceled

9. canceled

10. (original) A method as in claim 1, further comprising binding a translation of a return within the OL subroutine translation to an index in the return target cache.

11. (original) A method as in claim 10, further comprising:
setting a specified entry of the return target cache to a predetermined value indicating a lack of binding; and
upon sensing attempted access to the specified entry of the return target cache, scanning the return target cache and associating with the current unbound launch block an array index other than the specified index.

12. (original) A method for implementing subroutine calls and returns in a computer system comprising the following steps:
A) converting a sequence of input language (IL) instructions of a guest system into a corresponding sequence of output language (OL) instructions of a host system;
B) executing the OL instructions in the host system;
C) for each call to an IL subroutine made from any of a plurality of IL call sites in the IL instruction sequence:
i) translating each IL call site into a corresponding OL call site;
ii) storing a call site IL return address R_{call} on a stack;
iii) calculating a first index by evaluating a function with P as an argument, where P is a procedure entry address of the subroutine;
iv) storing a corresponding OL return address R' in a return target

cache at a location determined by the first index, the return target cache comprising an array of elements;

v) executing an OL subroutine translation of the called IL subroutine;

D) upon completion of execution of the OL subroutine translation,

i) retrieving an OL target address from the return target cache at the location indicated by a second index; and

ii) continuing execution beginning at the OL target address.

E) generating a confirm block of instructions corresponding to each OL call site and, upon execution of any confirm block of instructions:

i) comparing an actual IL return target address R_{actual} fetched from the stack with the predicted IL return address R_{pred};

ii) if R_{actual} is equal to R_{pred}, continuing execution of the OL instructions following the OL call site; and

iii) if R_{actual} is not equal to R_{pred}, transferring execution to the back-up return address recovery module; and

F) in the back-up return address recovery module, determining a correct OL return address.

13. (original) A method as in claim 12, further comprising binding a translation of a return within the OL subroutine translation to an index in the return target cache.

14. (original) A system for implementing subroutine calls and returns in a computer system comprising:

A) a host computer system that executes host instructions in an output language OL;

B) a guest system that is operatively connected to the host system and that issues a sequence of instructions in an input language (IL) including a call to a subroutine;

C) a binary translator converting the sequence of input language (IL) instructions of the guest system into a corresponding sequence of the output language (OL) instructions of the host system and storing the OL instructions in a translation cache,

D) the binary translator comprising computer-executable instructions for translating an IL subroutine call and an IL subroutine return into corresponding OL instruction sequences, including a call block and a launch block of OL instructions;

E) the call block, upon each call to an IL subroutine from an IL call site in the IL instruction sequence, comprising computer-executable instructions

i) for storing a call site IL return address R_{call} of the call on a stack;

ii) for determining a first index by evaluating a function with P as an argument, where P is a procedure entry address of the subroutine; and

iii) for storing a corresponding OL return address R' in a return target cache at a location determined by the first index;

iv) for transferring execution to the OL subroutine translation of the called IL subroutine;

F) the launch block, upon completion of execution of the OL subroutine translation, further comprising computer-executable instructions

i) for popping an actual IL return address R_{actual} from the stack;

ii) for retrieving an OL target address from the return target cache at the location indicated by a second index; and

iii) for continuing execution beginning at the OL target address.

15. (original) A system as in claim 14, in which:

there is a plurality of IL call sites;

the binary translator comprises computer-executable instructions

for translating each IL call site into a corresponding OL call site;

for inserting a confirm block of instructions into each OL call site;

for comparing R_{actual} with a predicted IL return address R_{pred} corresponding to the current OL call site;

for continuing execution of the OL instructions following the OL call site if R_{actual} is equal to R_{pred}; and

for transferring execution to the back-up return address recovery module if R_{actual} is not equal to R_{pred}.

16. (original) A system as in claim 14, in which the binary translator comprises further computer-executable instructions for binding a translation of a return within the OL subroutine translation to an index in the return target cache.

17. (new) A method for implementing subroutine calls and returns in a computer system comprising:

A) converting a sequence of input language (IL) instructions into a corresponding sequence of output language (OL) instructions;

B) executing the OL instructions;

C) for each call to an IL subroutine made from an IL call site in the IL instruction sequence:

i) storing a call site IL return address R_{call} on a stack;

ii) calculating a first index by evaluating a function with P as an argument, where P is a procedure entry address of the subroutine;

iii) storing a corresponding OL return address in a return target cache at a location indicated by the first index;

iv) executing an OL subroutine translation of the called IL subroutine;

D) upon completion of execution of the OL subroutine translation,

i) in a launch block of instructions, retrieving an OL target address from the return target cache at the location indicated by a second index; and

ii) continuing execution beginning at the OL target address;

in which:

the return target cache is an array having a plurality of elements;

the function maps IL procedure entry addresses substantially uniformly over the return target cache; and

each of the elements of the return target cache is identified by an array index, and the function extracts a number of bits from the address P.

18. (new) A method for implementing subroutine calls and returns in a computer system comprising:

A) converting a sequence of input language (IL) instructions into a corresponding sequence of output language (OL) instructions;

B) executing the OL instructions;

C) for each call to an IL subroutine made from an IL call site in the IL instruction sequence:

i) storing a call site IL return address R_{call} on a stack;

ii) calculating a first index by evaluating a function with P as an argument, where P is a procedure entry address of the subroutine;

iii) storing a corresponding OL return address in a return target cache at a location indicated by the first index;

iv) executing an OL subroutine translation of the called IL subroutine;

D) upon completion of execution of the OL subroutine translation,

i) in a launch block of instructions, retrieving an OL target address from the return target cache at the location indicated by a second index; and

ii) continuing execution beginning at the OL target address;

in which:

the step of calculating the first index k is performed as part of the step of converting the IL call into the corresponding sequence of OL instructions.